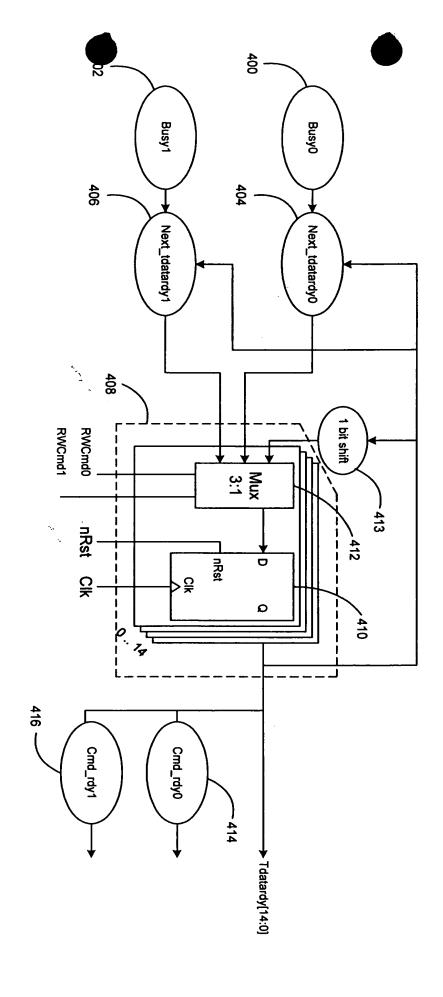


FIG. 3F



```
// SYNCHRONOUS MEMORY DATA HANDLER BLOCK
// TlengthO and Tlength1: Burst length - 1.
// Len0 and Len1: Based on Tlength with a value between 0 and 3.
// Trob0 and Trob1: Rest of burst. Indicates number of words until next column address
// Busy0 and Busy1: A bit stream that represents the number of burst words.
 // Tdatardy: The shift register.
 // Mask: A shift register to insert wait states between accesses.
 // Sync0 and Sync1: Indicates the access is for synchronous memory.
// TreadO and Tread1: Indicates the access is a read, otherwise it is a write.
// CsdO and Csdl: The initial access latency.
// BusyO and Busyl: A four bit vector of 1's.
 // MaskO and Mask1: A five bit vector of 1's.
 // Next_tdatardyO and Next_tdatardyl: Value to be synchronously loaded into Tdatardy.
 // Next mask0 and Next mask1: Value to be synchronously loaded into Mask shift register.
// Cmd rdy0 and Cmd rdy1: Tells master that it is okay to commit to the access. // Next rdy0 and Next rdy1: Logic to force a dead clock between accesses.
// Create length that decrements based on RWCmd.
 // Note: Tont decrements on Tdatardy in the target, which is much too late.
      always @(Tlength0)
      begin
         if (|Tlength0[4:2])
                                           // Length > 2'b11
          Len0 = 2'b11;
         else
          Len0 = Tlength0[1:0];
      end
// Create length that decrements based on RWCmd.
// Note: Tcnt decrements on Tdatardy in the target, which is much too late.
      always @(Tlength1)
      begin
         if (|Tlength1[4:2])
                                            // Length > 2'b11
          Len1 = 2'b11;
         else
          Len1 = Tlength1[1:0];
```

FIG. 5A

```
// Convert Trob into bit stream.
     always @(Trob0 or Len0)
     begin
                                     // synopsys full_case parallel_case
       case (Trob0)
                       Busy0[3:0] = 4'b0001;
       2'h0:
                                     // synopsys full_case parallel_case
       2'h1:
                case (Len0[1:0])
               2'h0: Busy0[3:0] = 4'b0001;
               2'h1: Busy0[3:0] = 4'b0011;
               2'h2: Busy0[3:0] = 4'b0011;
               2'h3: Busy0[3:0] = 4'b0011;
               endcase
               case (Len0[1:0])  // synopsys full_case parallel_case
2'h0: Busy0[3:0] = 4'b0001;
       2'h2:
               2'h1: Busy0[3:0] = 4'b0011;
                      Busy0[3:0] = 4'b0111;
               2'h2:
               2'h3: Busy0[3:0] = 4'b0111;
               endcase
       2'h3:
               case (Len0[1:0])
                                     // synopsys full_case parallel_case
               2'h0: Busy0[3:0] = 4'b0001;
               2'h1: Busy0[3:0] = 4'b0011;
2'h2: Busy0[3:0] = 4'b0111;
               2'h3: Busy0[3:0] = 4'b1111;
               endcase
       endcase
     end
```

FIG. 5B

```
// Convert Trob into bit stream.
// ************
     always @(Trobl or Len1)
     begin
                                    // synopsys full_case parallel_case
       case (Trob1)
       2'h0:
                       Busy1(3:0) = 4'b0001;
                case (Len1[1:0])
                                  // synopsys full_case parallel_case
       2'h1:
                      Busy1[3:0] = 4'b0001;
               2'h0:
                      Busy1{3:0} = 4'b0011;
               2'h1:
                      Busy1(3:0) = 4'b0011;
               2'h2:
                      Busy1(3:0) = 4'b0011;
               2'h3:
               endcase
                                   // synopsys full_case parallel_case
       2'h2:
               case (Len1[1:0])
               2'h0:
                      Busy1[3:0] = 4'b0001;
               2'h1:
                      Busy1(3:0) = 4'b0011;
                      Busy1[3:0] = 4'b0111;
               2'h2:
                      Busy1[3:0] = 4'b0111;
               2'h3:
               endcase
               case (Len1[1:0]) // synopsys full_case parallel_case
2'h0: Busy1[3:0] = 4'b0001;
2'h1: Busy1[3:0] = 4'b0011;
2'h2: Busy1[3:0] = 4'b0111;
       2'h3:
               2'h3: Busy1(3:0) = 4'b1111;
               endcase
       endcase
     end
// Shift right by 1. Shift in zero for MSB.
// FIX: the high order bit should be shifted in as zero in the Next_tdatardy block.
     always @(posedge Clk or negedge nRst)
     begin
       if (!nRst)
       begin
         Tdatardy <= 15'h0000;
                  <= 15'h0000;
        Mask
       end
       else if (RWCmd0 & !RWCmd1)
       begin
                                                          FIG. 5C
        Tdatardy[13:0] <= Next_tdatardy0;</pre>
        Tdatardy[14]
                        <= 0;
        Mask[13:0]
                         <= Next_mask0;
        Mask[14]
                         <= 0;
       end
       else if (RWCmd1 & !RWCmd0)
       begin
        Tdatardy[13:0] <= Next_tdatardy1;
        Tdatardy[14]
                        <= 0;
                                             COPYRIGHT 1999 HEWLETT PACKARD COMPANY
        Mask[13:0]
                         <= Next_mask1;
        Mask[14]
                         <= 0;
       end
               // shift
       else
       begin
        Tdatardy[13:0] <= Tdatardy[14:1];
        Tdatardy[14]
```

```
Mask[13:0]
                       <= Mask[14:1];
        Mask[14]
                        <= 0;
       end
     end
// Generate next bit stream by concatenating old bit stream with new bit stream.
// Bit stream includes a right shift of one bit.
     always @(Tdatardy or Csd0 or Busy0 or Sync0 or Tread0 or Mask0)
       if (Sync0 && Tread0)
                                      // Read
      begin
                                       // synopsys parallel_case
        casex (Csd0)
        3'h1: begin
                Next_tdatardy0 = {6'h0, Busy0, Tdatardy[4:1]};
                Next_mask0
                              = {5'h0, Mask0, Tdatardy[4:1]};
              end
        3'h2: begin
                Next_tdatardy0 = {5'h0, Busy0, Tdatardy[5:1]};
                              = {4'h0, Mask0, Tdatardy[5:1]};
                Next_mask0
              end
        3'h3: begin
               Next_tdatardy0 = {4'h0,Busy0,Tdatardy[6:1]};
                             = {3'h0, Mask0, Tdatardy[6:1]};
                Next_mask0
        3'h4: begin
                Next tdatardy0 = {3'h0, Busy0, Tdatardy(7:1)};
                             = {2'h0, Mask0, Tdatardy [7:1]};
               Next_mask0
              end
        3'h5: begin
                Next_tdatardy0 = {2'h0,Busy0,Tdatardy[8:1]};
                Next_mask0
                              = {1'h0, Mask0, Tdatardy[8:1]};
              end
        3'h6: begin
                Next_tdatardy0 = {1'h0,Busy0,Tdatardy[9:1]};
               Next mask0
                             = {Mask0, Tdatardy[9:1]};
              end
        default: begin
                  Next_tdatardy0 = 14'bxx_xxxx_xxxx_xxxx;
                                                                   FIG. 5D
                  Next_mask0
                                 = 14'bxx_xxxx_xxxx_xxxx;
                 end
        endcase
       end
      else if (Sync0 && !Tread0)
                                      // Write
        Next_tdatardy0 = {9'h00,Busy0,Tdatardy{1}};
                       - (8'h00, Mask0, Tdatardy[1]);
        Next_mask0
      end
       else
                                      // async
      begin
        Next_tdatardy0 = 14'h0000;
                       = 14'h0000;
        Next_mask0
       end
     end
```

```
// Determine if new access can issue R/W command.
     always @(Tdatardy or Mask or Csdl or Syncl or Tread1)
     begin
      if (!Sync1)
       begin
        Cmd rdy1
                        = ~(|(Mask[14:0]));
                        = \sim (|(Mask[14:0]));
        Next_rdyl
       end
                                                         // Read
       else if (Tread1)
       begin
        case (Csdl)
                                                         // synopsys parallel_case
        3'h1: begin
                Cmd_rdy1 = \sim(|(Tdatardy[14:5]));
                Next_rdy1 = ~(|(Mask[14:5]));
              end
        3'h2: begin
                Cmd_rdy1 = \sim (|(Tdatardy[14:6]));
                Next_rdy1 = ~(|(Mask(14:6)));
              end
        3'h3: begin
                Cmd_rdy1 = ~(|(Tdatardy[14:7]));
                Next_rdy1 = ~(|(Mask(14:7]));
              end
        3'h4: begin
                Cmd_rdy1 = -(|(Tdatardy[14:8]));
                Next_rdyl = -(|(Mask[14:8]));
              end
        3'h5: begin
                Cmd_rdy1 = ~(|(Tdatardy[14:9]));
                Next_rdy1 = ~(|(Mask[14:9]));
              end
        3'h6: begin
                Cmd_rdy1 = ~(|(Tdatardy[14:10]));
                Next_rdy1 = ~(| (Mask[14:10]));
              end
        default: begin
                                                                      FIG. 5E
                Cmd_rdy1 = ~(|(Tdatardy[14:5]));
Next_rdy1 = ~(|(Mask[14:5]));
                 end
        endcase
      end
                                                         // Write
      else
      begin
        Cmd_rdy1 = ~(|(Tdatardy[14:2]));
        Next_rdy1 = ~(|(Mask[14:2]));
      end
    end
```

```
// Determine if new access can issue R/W command.
    always @(Tdatardy or Mask or Csd0 or Sync0 or Tread0)
    begin
       if (!Sync0)
       begin
        Cmd_rdy0
                         = \sim (|(Mask[14:0]));
                        = ~(|(Mask{14:0]));
        Next_rdy0
       end
                                           // Read
       else if (Tread0)
       begin
                                          // synopsys parallel_case
        case (Csd0)
        3'h1: begin
                Cmd_rdy0 = ~(|(Tdatardy{14:5}));
                Next_rdy0 = -(|(Mask[14:5]));
               end
        3'h2: begin
                Cmd_rdy0 = ~(|(Tdatardy[14:6]));
                Next_rdy0 = ~(|(Mask[14:6]));
               end
        3'h3: begin
                Cmd_rdy0 = \sim([(Tdatardy[14:7]));
                Next_rdy0 = ~([(Mask[14:7]));
               end
        3'h4: begin
                Cmd_rdy0 = ~(|(Tdatardy[14:8]));
Next_rdy0 = ~(|(Mask[14:8]));
        3'h5: begin
                Cmd rdy0 = \sim (|(Tdatardy[14:9]));
                Next_rdy0 = ~(|(Mask[14:9]));
        3'h6: begin
                Cmd_rdy0 = ~(((Tdatardy[14:10]));
                Next_rdy0 = ~(| (Mask[14:10]));
               end
        default: begin
                Cmd_rdy0 = ~(|(Tdatardy[14:5]));
Next_rdy0 = ~(|(Mask[14:5]));
                 end
        endcase
                                                                        FIG. 5G
       end
                                           // Write
       else
      begin
        Cmd_rdy0 = ~(|(Tdatardy[14:2]));
        Next_rdy0 = ~(|(Mask[14:2]));
       end
    end
```

```
// Generate next bit stream by concatenating old bit stream with new bit stream.
// Bit stream includes a right shift of one bit.
     always @(Tdatardy or Csdl or Busyl or Syncl or Treadl or Maskl)
    begin
      if (Syncl && Tread1)
                                                              // Read
      begin
        casex (Csd1)
                                                              // synopsys parallel_case
        3'h1: begin
                Next_tdatardyl = {6'h0, Busyl, Tdatardy{4:1}};
                Next_mask1
                               = {5'h0, Mask1, Tdatardy[4:1]};
              end
        3'h2: begin
                Next tdatardyl = (5'h0, Busyl, Tdatardy[5:1]);
                              = (4'h0, Mask1, Tdatardy [5:1]);
                Next_maskl
              end
        3'h3: begin
                Next_tdatardy1 = {4'h0, Busy1, Tdatardy[6:1]};
                             = (3'h0, Mask1, Tdatardy [6:1]);
                Next mask1
              end
        3'h4: begin
                Next_tdatardy1 = (3'h0, Busy1, Tdatardy[7:1]);
                              = {2'h0, Mask1, Tdatardy [7:1]};
                Next_mask1
              end
        3'h5: begin
                Next_tdatardy1 = {2'h0, Busy1, Tdatardy[8:1]};
                Next_mask1
                               = {1'h0, Mask1, Tdatardy [8:1]};
              end
        3'h6: begin
                Next_tdatardy1 = {1'h0, Busy1, Tdatardy[9:1]};
                             = (Mask1, Tdatardy [9:1]);
                Next_mask1
              end
        default: begin
                   Next_tdatardyl = 14'bxx_xxxx_xxxx_xxxx;
                   Next mask1
                                 = 14'bxx_xxxx_xxxx_xxx;
                 end
        endcase
      end
      else if (Syncl && !Tread1)
                                                             // Write
      begin
        Next_tdatardy1 = {9'h00,Busy1,Tdatardy[1]};
                       = {8'h00, Mask1, Tdatardy[1]);
        Next_mask1
      end
      else
                                                             // Async
      begin
        Next_tdatardy1 = 14'h0000;
                                                                            FIG. 5F
        Next_mask1
                       - 14'h0000;
      end
    end
```